User Interface (UI) Design

using

Flutter

1a. Install Flutter and Dart SDK

Installing on Windows:

Before start installation you need to make sure that your system supports minimum hardware requirements.

System Requirements:

- ➪ Windows 10 or 11 (64-bit)
- ➪ Disk Space: 1.64 GB (does not include IDE/tools)
- ➪ Command Line Tools: PowerShell 5.0 or newer
- ➪ Git for Windows (optional)

Getting Flutter SDK:

1. Download flutter manually using link :
https://storage.googleapis.com/flutter_infra_release/releases/stable/windows/flutter_windows_3.32.4-stable.zip
2. Extract the File in desired location
3. Update the path permanently (the following is for Windows 10):
   a. Open Control Panel and drill down to Desktop App ➪ User Accounts ➪ User Accounts ➪ Change My Environment Variables.
4. In the Edit environment variable, click the New button.
   ii. Type the path Ex: C:\Users\WindowsUserName\flutter\bin.
   iii. Click the OK button and then close the Environment Variables screen.

Verify Flutter Installation

1. Open Command Prompt or PowerShell.
2. Run: flutter doctor
3. This checks dependencies and shows you what needs to be fixed (example: installing Android Studio, SDKs, etc.)

Install Dart Plugin

- You don't need to install Dart separately — it comes bundled with Flutter.
- When you install Flutter, Dart SDK is included in flutter/bin/cache/dart-sdk.

Install an IDE (Optional but recommended)

- Visual Studio Code → install Flutter and Dart extensions.(Ctrl+ Shift +X)

OR

- Android Studio → install Flutter and Dart plugins from Preferences → Plugins.

1b. Write a simple Dart program to understand the language basics.

```
void main() {
  //  Variables & Data Types
  String name = 'Uday';
  int age = 31;
  double height = 5.8;
  bool isTeacher = true;

  //  Print variables
  print('Name: $name');
  print('Age: $age');
  print('Height: $height');
  print('Is Teacher: $isTeacher');

  // List (Array)
  List<String> subjects = ['Dart', 'Flutter', 'AI', 'Python'];

  //  Loop through the list
  print('\nSubjects Known:');
  for (String subject in subjects) {
    print('- $subject');
  }

  //  Function call
  greetUser(name);
}

//  Function definition
void greetUser(String userName) {
  print('\nWelcome, $userName! Enjoy learning Dart.');
}
```

```
Age: 31
Height: 5.8
Is Teacher: true

Subjects Known:
- Dart
- Flutter
- AI
- Python

Welcome, Uday! Enjoy learning Dart.
```

⇨ **Steps to run above code:**

Step 1: Open VS Code software

(Make sure Flutter & Dart Extensions are installed)

Step 2: Navigate to File Menu → New File → in search bar type your desired program name   Ex: main.dart

Step 3: Save program in your desired folder

Step 4: Go to Terminal → New Terminal

Step 5: in Powershell's Terminal → dart Desired progarm name

Ex: dart main.dart

Note:

- without Flutter SDK installation locally in your computer you won't be able to use flutter and dart commands internal
- VS Code might show error like "Flutter SDK not Fund"

2a. Explore various Flutter widgets (Text, Image, Container, etc.).

Steps:

Step 1: Open VS code

Step 2: ctrl+ shift + P → type and select Flutter: New Project

Step 3: choose Flutter Application

Step 4: Enter project name ex: Flutter_Application1

Step 5: choose folder location and wait for project creation to complete

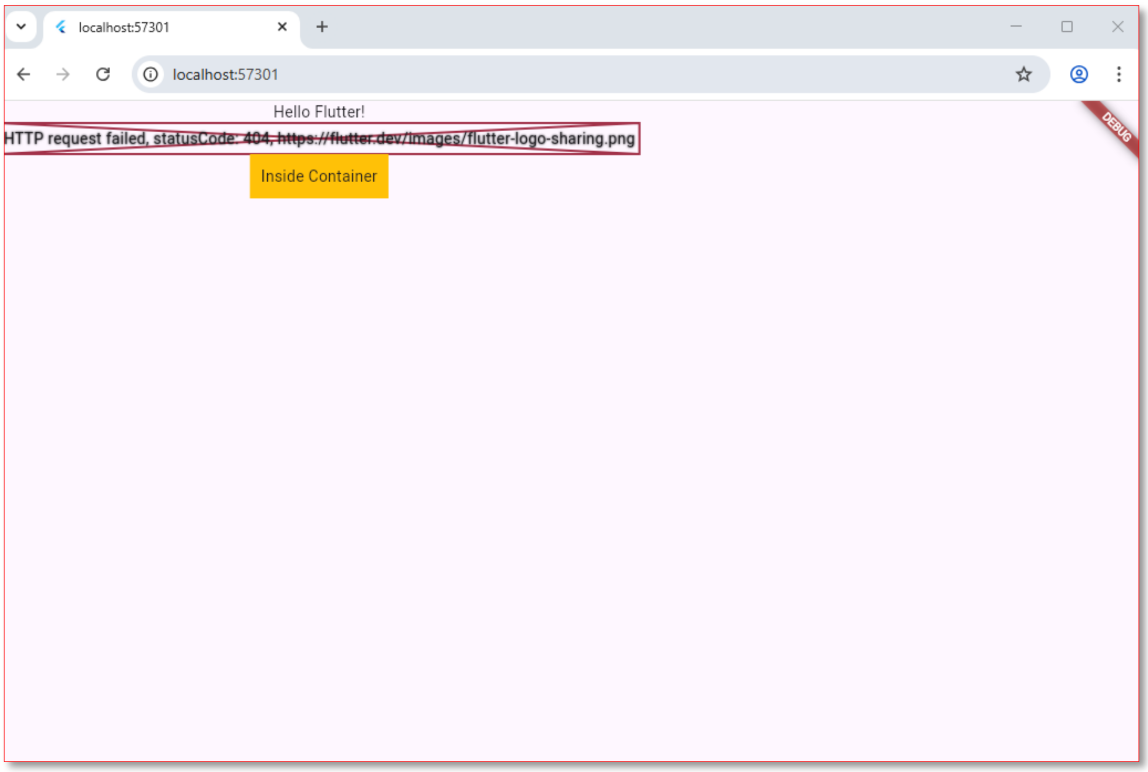
Step 6: open lib/main.dart and delete existing code and type below:

```
import 'package:flutter/material.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
   return MaterialApp(
     home: Scaffold(
       body: Column(
         children: [
           Text('Hello Flutter!'),
           Image.network(
             'https://flutter.dev/images/flutter-logo-sharing.png',
             height: 80,
           ),
           Container(
             padding: EdgeInsets.all(10),
             color: Colors.amber,
             child: Text('Inside Container'),
           ),
         ],
       ),
     ),
   );
  }
}
```

Step 7: Bottom right corner choose device: ex: edge /chrome / android emulator

Step 8: press F5 or run→ start debugging or in terminal use command "flutter run"(recommended).

Output:


localhost:57301

Hello Flutter!

~~HTTP request failed, statusCode: 404, https://flutter.dev/images/flutter-logo-sharing.png~~

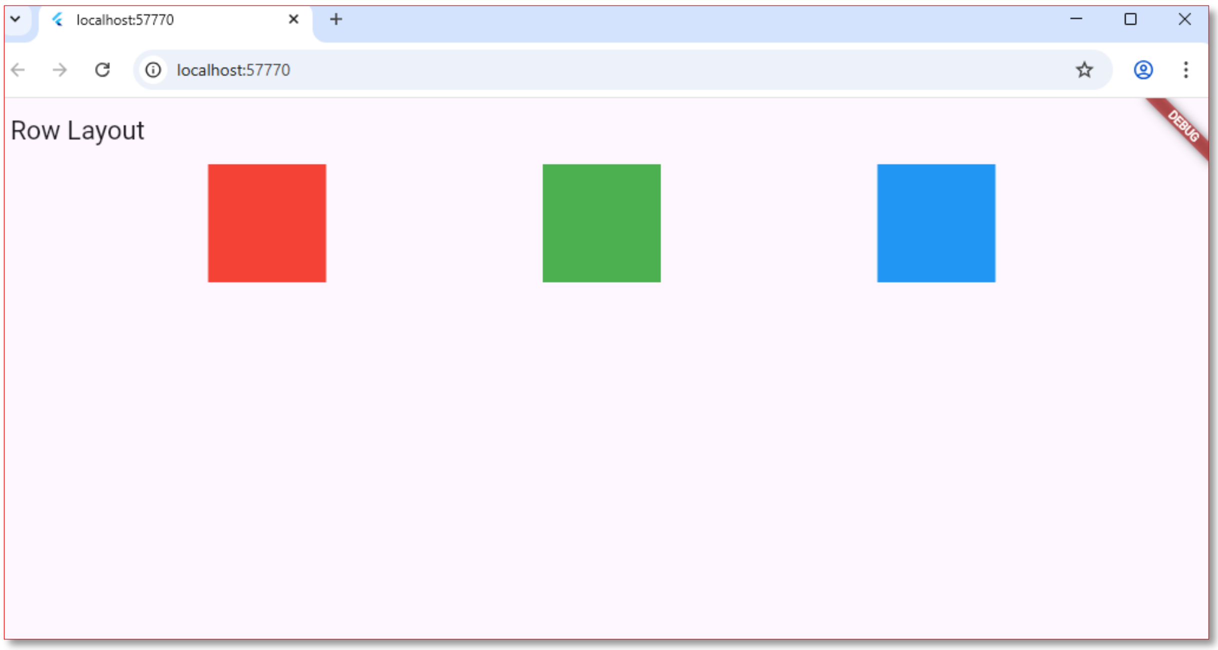
Inside Container

2b. Implement different layout structures using Row, Column, and Stack widgets.

Row Layout:

```
import 'package:flutter/material.dart';

void main() {
 runApp(MyApp());
}

class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return MaterialApp(
   home: Scaffold(
    appBar: AppBar(title: Text('Row Layout')),
    body: Row(
     mainAxisAlignment: MainAxisAlignment
        .spaceEvenly, // Spaces children evenly in the row
     children: <Widget>[
          Container(color: Colors.red, width: 100, height: 100),
          Container(color: Colors.green, width: 100, height: 100),
          Container(color: Colors.blue, width: 100, height: 100),
        ],
      ),
    ),
  );
 }
}
```

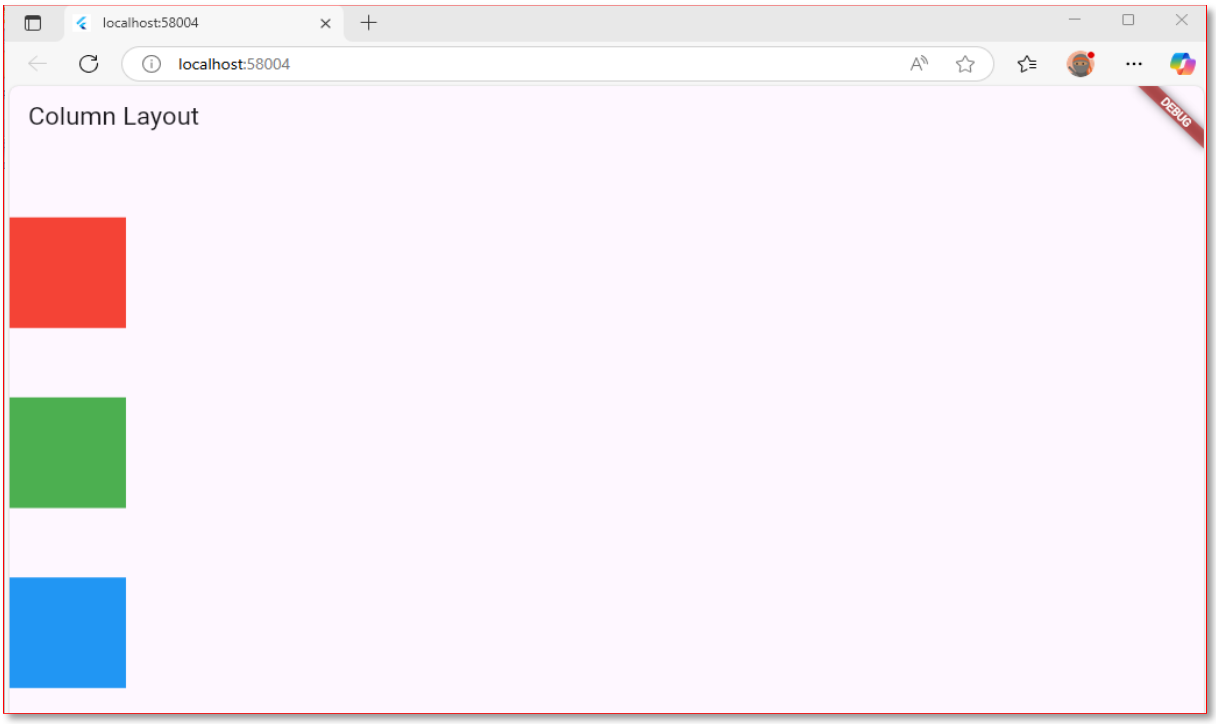
Output:



Row Layout

Column Layout:

```
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(title: Text('Column Layout')),
        body: Column(
          mainAxisAlignment: MainAxisAlignment
              .spaceEvenly, // Spaces children evenly in the column
          children: <Widget>[
            Container(color: Colors.red, width: 100, height: 100),
            Container(color: Colors.green, width: 100, height: 100),
            Container(color: Colors.blue, width: 100, height: 100),
          ],
        ),
      ),
    );
  }
}
```

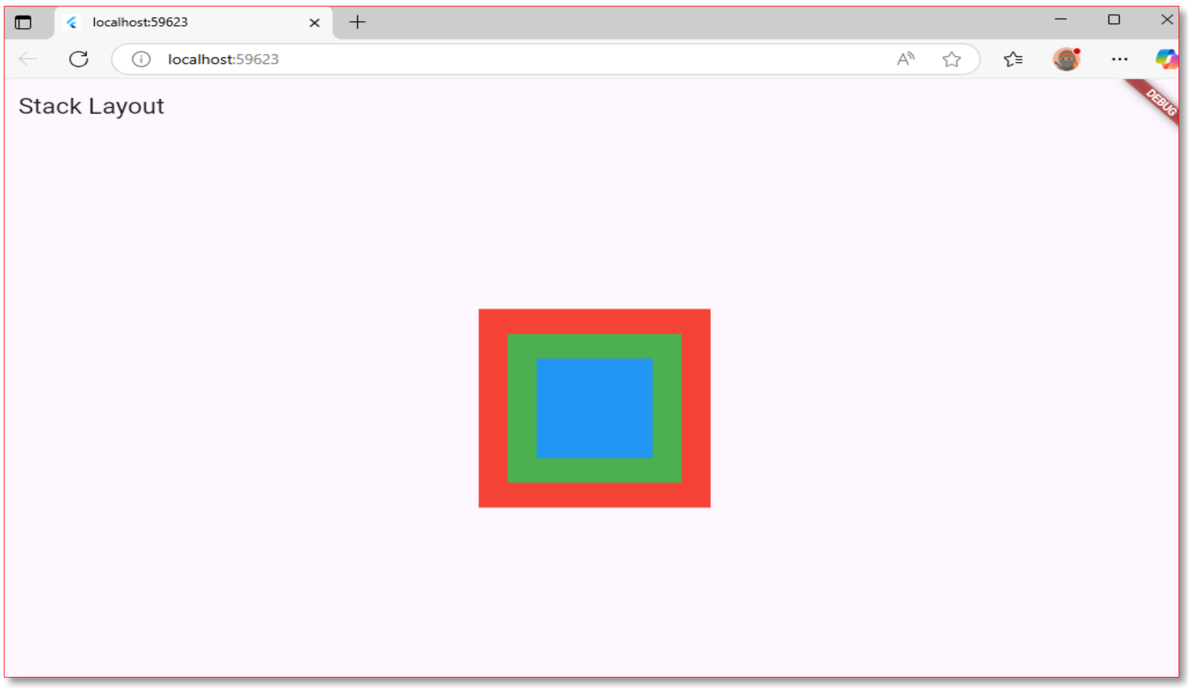


Column Layout

Stack widget:

```
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key); // ✅Added Key

  @override
  Widget build(BuildContext context) {
   return MaterialApp(
     home: Scaffold(
       appBar: AppBar(title: const Text('Stack Layout')),
        body: Center(
          child: Stack(
            alignment: Alignment.center,
            children: <Widget>[
              Container(color: Colors.red, width: 200, height: 200),
              Container(color: Colors.green, width: 150, height: 150),
              Container(color: Colors.blue, width: 100, height: 100),
            ],
          ),
        ),
      ),
    );
  }
}
```

Output:

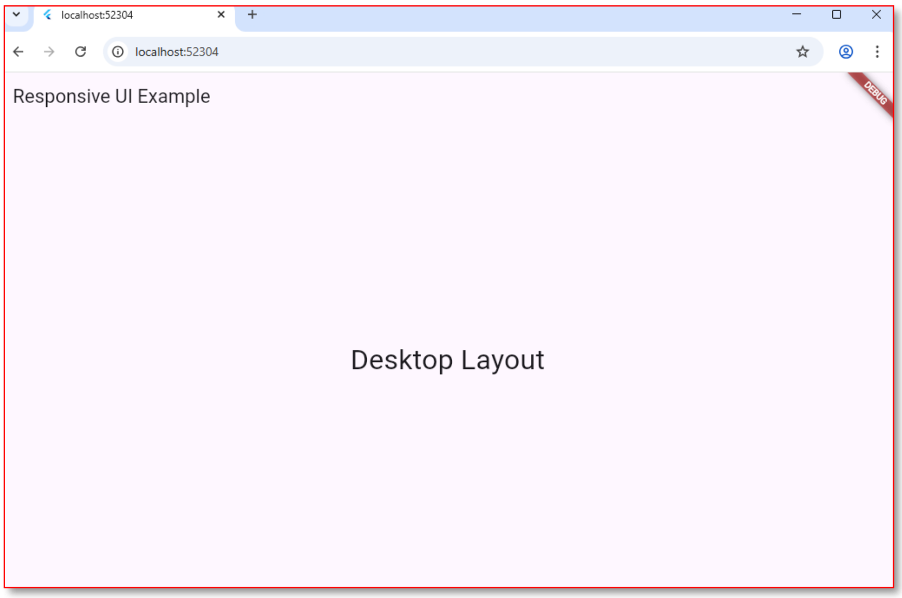


Stack Layout

3A) Design a responsive UI that adapts to different screen sizes.

```
import 'package:flutter/material.dart';

void main() => runApp(const MyApp());

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
   return MaterialApp(
     home: Scaffold(
       appBar: AppBar(title: const Text("Responsive UI Example")),
       body: LayoutBuilder(
         builder: (context, constraints) {
           if (constraints.maxWidth < 600) {
             return const Center(
               child: Text("Mobile Layout", style: TextStyle(fontSize: 24)),
             );
           } else {
             return const Center(
               child: Text("Desktop Layout", style: TextStyle(fontSize: 32)),
             );
           }
         },
       ),
     ),
   );
  }
}
```

Output:



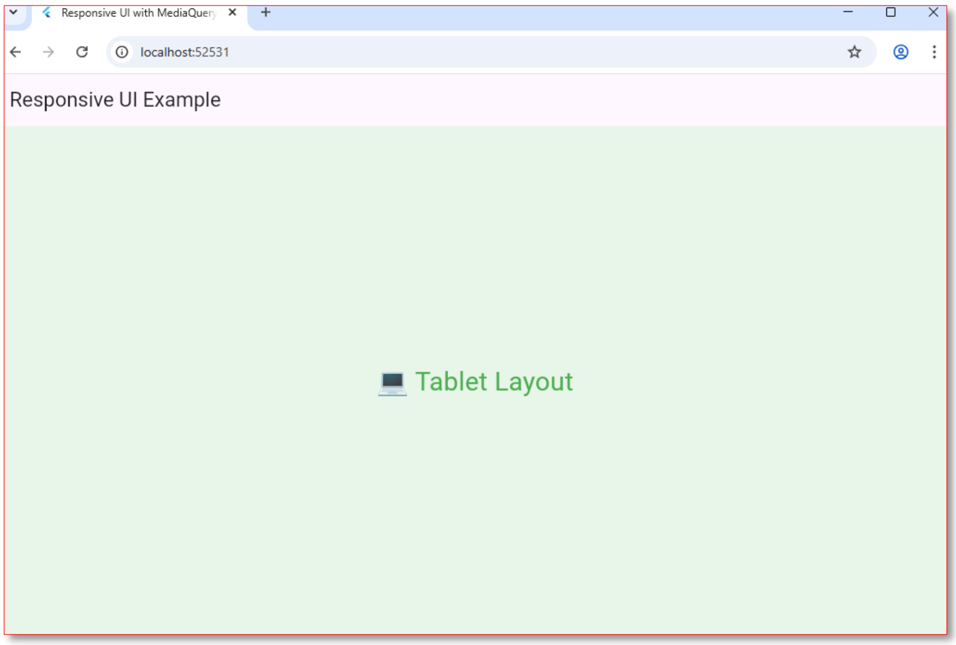
Responsive UI Example

Desktop Layout

3. b) Implement media queries and breakpoints for responsiveness.

```
import 'package:flutter/material.dart';

void main() => runApp(const MyApp());

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
   return MaterialApp(
     title: 'Responsive UI with MediaQuery',
     debugShowCheckedModeBanner: false,
     home: Scaffold(
       appBar: AppBar(title: const Text("Responsive UI Example")),
       body: const ResponsiveLayout(),
     ),
   );
  }
}

class ResponsiveLayout extends StatelessWidget {
  const ResponsiveLayout({super.key});

  @override
  Widget build(BuildContext context) {
   // Get screen width
   double screenWidth = MediaQuery.of(context).size.width;

   // Apply breakpoints
   if (screenWidth < 600) {
     return _buildMobileLayout();
   } else if (screenWidth < 1200) {
     return _buildTabletLayout();
   } else {
     return _buildDesktopLayout();
   }
 }

 // Mobile layout
 Widget _buildMobileLayout() {
   return Container(
     color: Colors.blue.shade50,
     child: const Center(
       child: Text(
         " Mobile Layout",
         style: TextStyle(fontSize: 24, color: Colors.blue),
       ),
     ),
   );
 }

 // Tablet layout
 Widget _buildTabletLayout() {
   return Container(
     color: Colors.green.shade50,
     child: const Center(
       child: Text(
         " Tablet Layout",
         style: TextStyle(fontSize: 28, color: Colors.green),
       ),
     ),
   );
 }

 // Desktop layout
 Widget _buildDesktopLayout() {
   return Container(
     color: Colors.red.shade50,
     child: const Center(
       child: Text(
         " Desktop Layout",
          style: TextStyle(fontSize: 32, color: Colors.red),
        ),
      ),
    );
  }
}
```

Emojis like computer , tablet, mobile symbols → no special library is needed — these are just **built-in emojis**, part of Unicode, and Flutter supports them out of the box.

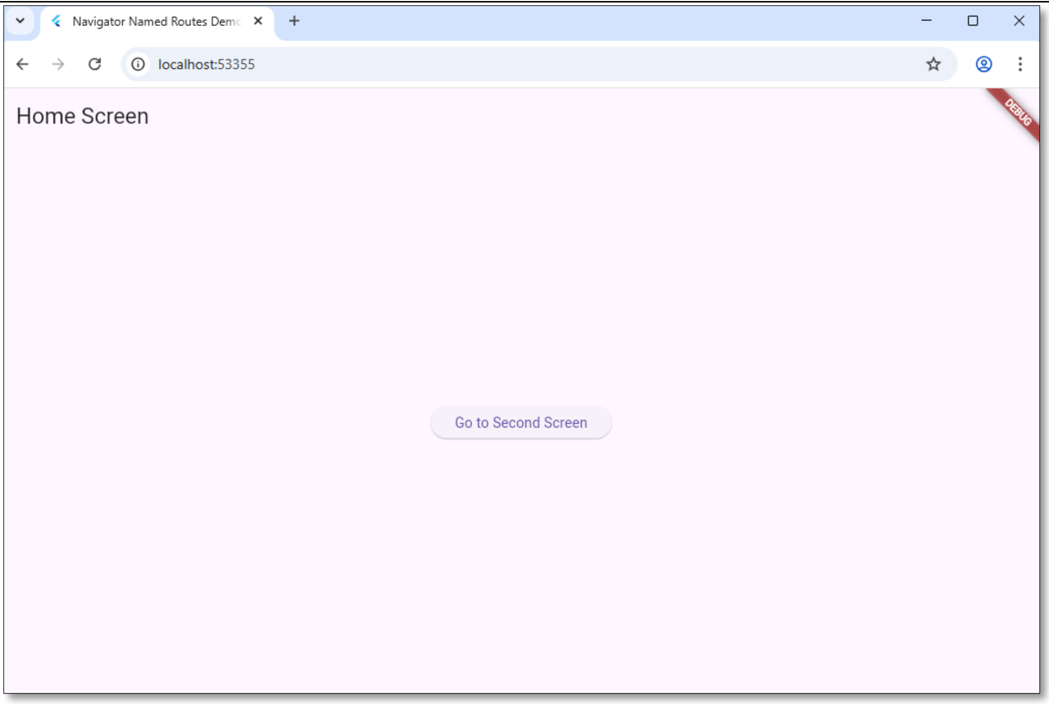


Responsive UI Example

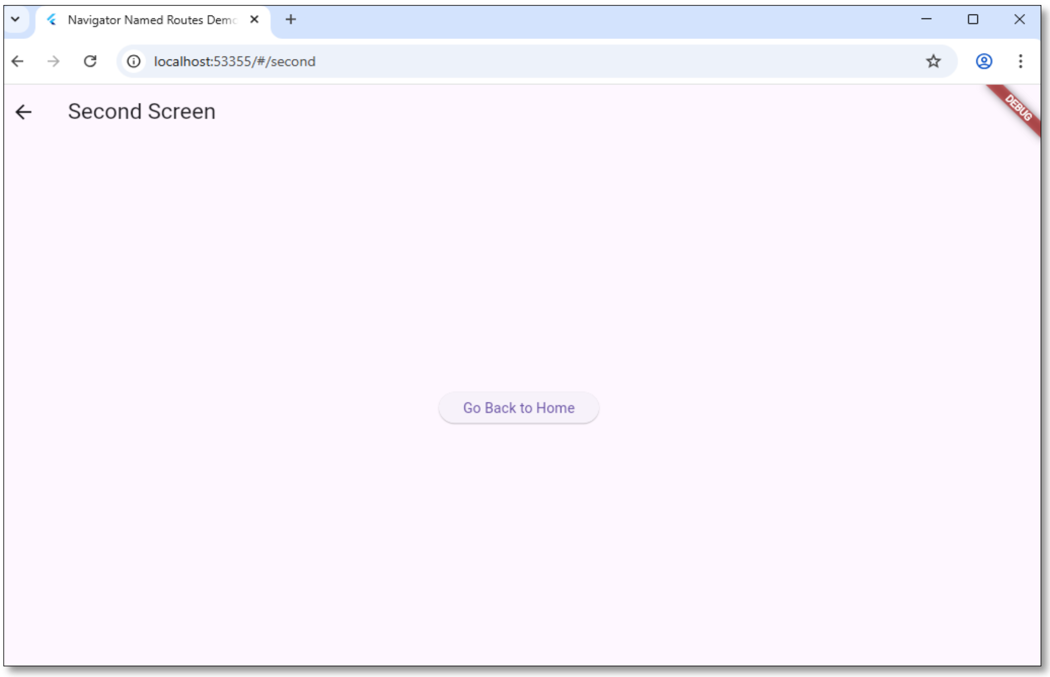
💻 Tablet Layout

4a) Set up navigation between different screens using Navigator.

```
import 'package:flutter/material.dart';

void main() {
 runApp(NavigationDemoApp());
}

class NavigationDemoApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return MaterialApp(
   title: 'Navigator Named Routes Demo',
   initialRoute: '/',  // default route
   routes: {
    '/': (context) => HomeScreen(),
    '/second': (context) => SecondScreen(),
   },
  );
 }
}

class HomeScreen extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(title: Text('Home Screen')),
   body: Center(
    child: ElevatedButton(
     child: Text('Go to Second Screen'),
     onPressed: () {
      // Navigate to second screen using route name
      Navigator.pushNamed(context, '/second');
     },
    ),
   ),
  );
 }
}

class SecondScreen extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(title: Text('Second Screen')),
   body: Center(
    child: ElevatedButton(
     child: Text('Go Back to Home'),
     onPressed: () {
          // Go back to previous screen
          Navigator.pop(context);
        },
      ),
    ),
  );
 }
}
```



Home Screen

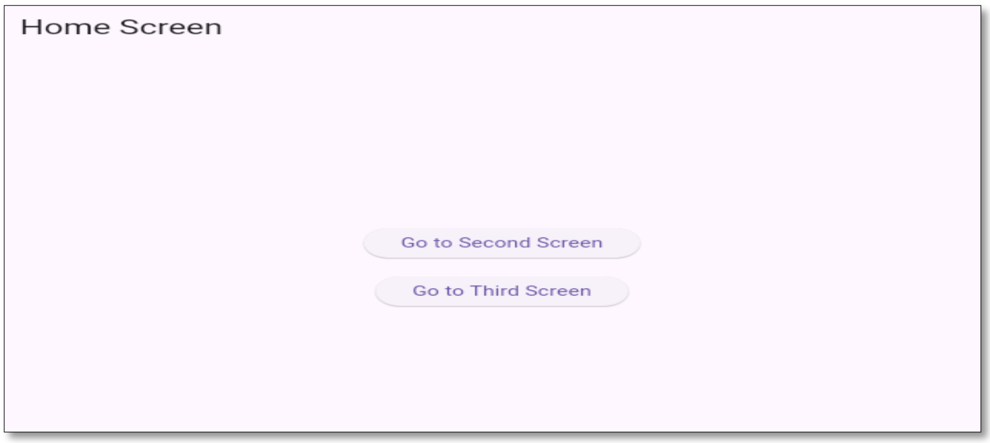


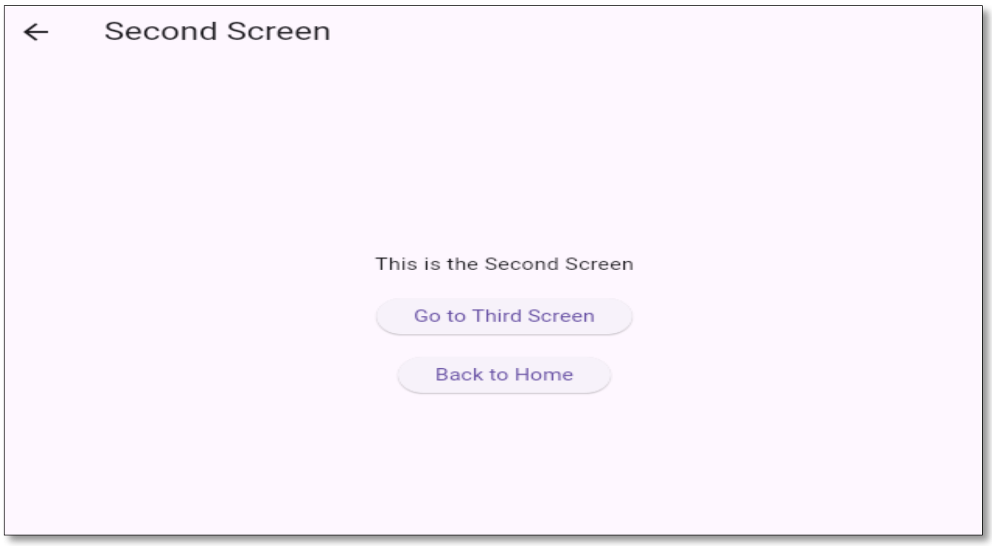
Second Screen

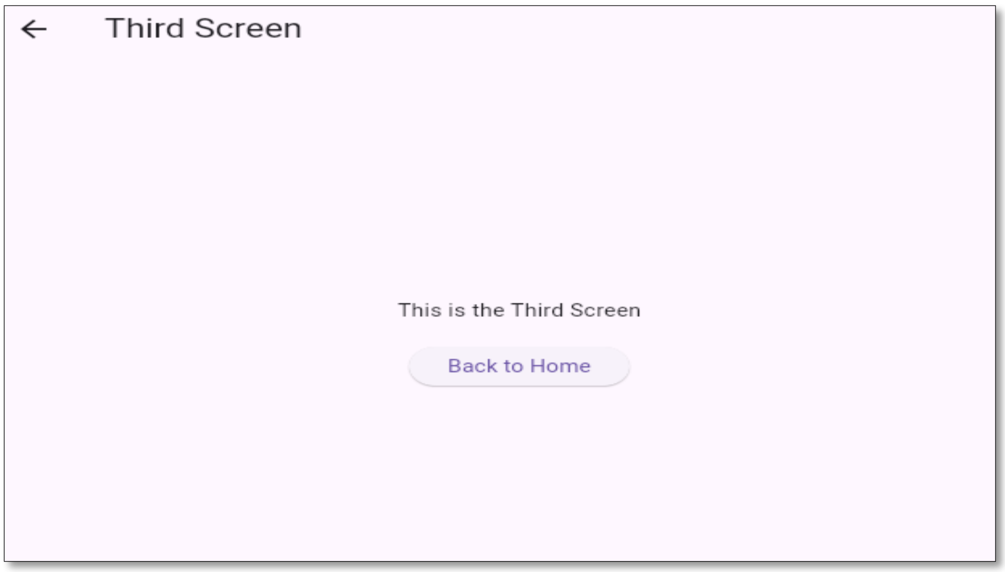
4b) Implement navigation with named routes.

```
import 'package:flutter/material.dart';

void main() {
  runApp(NamedRoutesApp());
}

class NamedRoutesApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Named Routes Demo',
      debugShowCheckedModeBanner: false,

      // First page that opens
      initialRoute: '/',

      // Define the routes
      routes: {
        '/': (context) => HomeScreen(),
        '/second': (context) => SecondScreen(),
        '/third': (context) => ThirdScreen(),
      },
    );
  }
}

class HomeScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Home Screen')),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            ElevatedButton(
              child: Text('Go to Second Screen'),
              onPressed: () {
                Navigator.pushNamed(context, '/second');
              },
            ),
            SizedBox(height: 20),
            ElevatedButton(
              child: Text('Go to Third Screen'),
              onPressed: () {
```

```
              Navigator.pushNamed(context, '/third');
            },
          ),
        ],
      ),
    ),
  );
 }
}

class SecondScreen extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(title: Text('Second Screen')),
    body: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Text('This is the Second Screen'),
          SizedBox(height: 20),
          ElevatedButton(
            child: Text('Go to Third Screen'),
            onPressed: () {
              Navigator.pushNamed(context, '/third');
            },
          ),
          SizedBox(height: 20),
          ElevatedButton(
            child: Text('Back to Home'),
            onPressed: () {
              Navigator.popUntil(context, ModalRoute.withName('/'));
            },
          ),
        ],
      ),
    ),
  );
 }
}

class ThirdScreen extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(title: Text('Third Screen')),
    body: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Text('This is the Third Screen'),
          SizedBox(height: 20),
          ElevatedButton(
            child: Text('Back to Home'),
            onPressed: () {
              Navigator.popUntil(context, ModalRoute.withName('/'));
            },
          ),
        ],
      ),
    ),
  );
 }
}
```



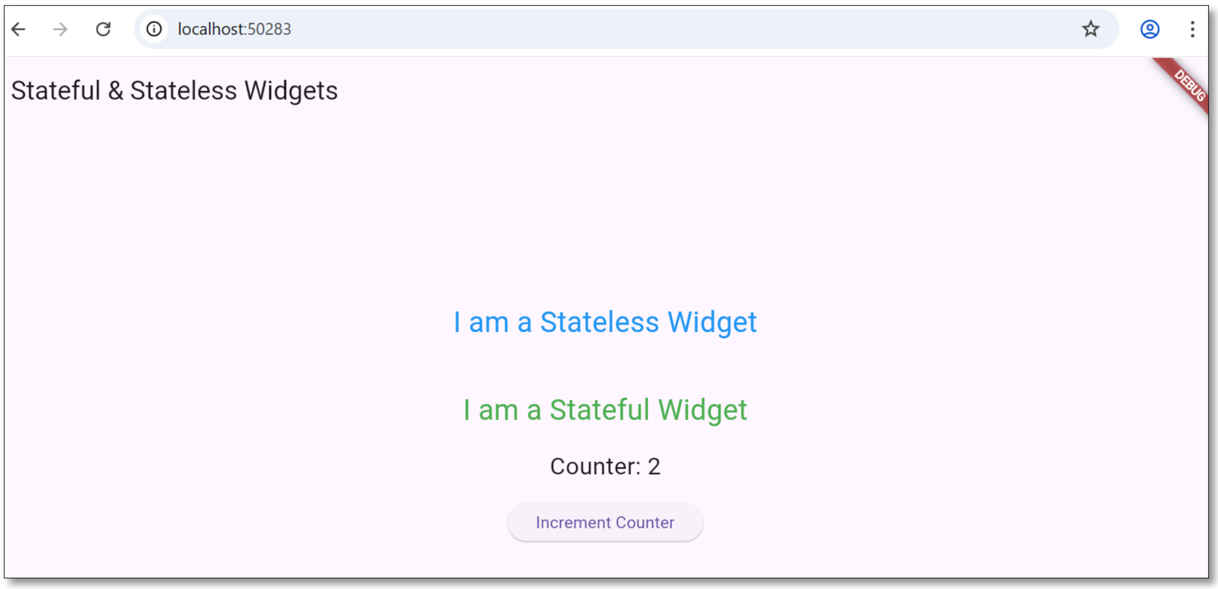




This is the Third Screen

5a) Learn about stateful and stateless widgets.

```
import 'package:flutter/material.dart';

void main() {
  runApp(StateDemoApp());
}

class StateDemoApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Stateful & Stateless Demo',
      home: Scaffold(
        appBar: AppBar(
          title: Text('Stateful & Stateless Widgets'),
        ),
        body: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            // Stateless widget example
            MyStatelessWidget(),

            SizedBox(height: 40),

            // Stateful widget example
            MyStatefulWidget(),
          ],
        ),
      ),
    );
  }
}

// Stateless widget: does NOT hold state
class MyStatelessWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Center(
      child: Text(
        'I am a Stateless Widget',
        style: TextStyle(fontSize: 24, color: Colors.blue),
      ),
    );
  }
}

// Stateful widget: holds state (counter in this example)
class MyStatefulWidget extends StatefulWidget {
  @override
  _MyStatefulWidgetState createState() => _MyStatefulWidgetState();
}

class _MyStatefulWidgetState extends State<MyStatefulWidget> {
  int _counter = 0;

  void _incrementCounter() {
    setState(() {
      _counter++; // updates the state
    });
  }

  @override
  Widget build(BuildContext context) {
    return Center(
      child: Column(
        children: [
          Text(
            'I am a Stateful Widget',
            style: TextStyle(fontSize: 24, color: Colors.green),
          ),
          SizedBox(height: 16),
          Text(
            'Counter: $_counter',
            style: TextStyle(fontSize: 20),
          ),
          SizedBox(height: 16),
          ElevatedButton(
            onPressed: _incrementCounter,
            child: Text('Increment Counter'),
          ),
        ],
      ),     a
    );
  }
}
```



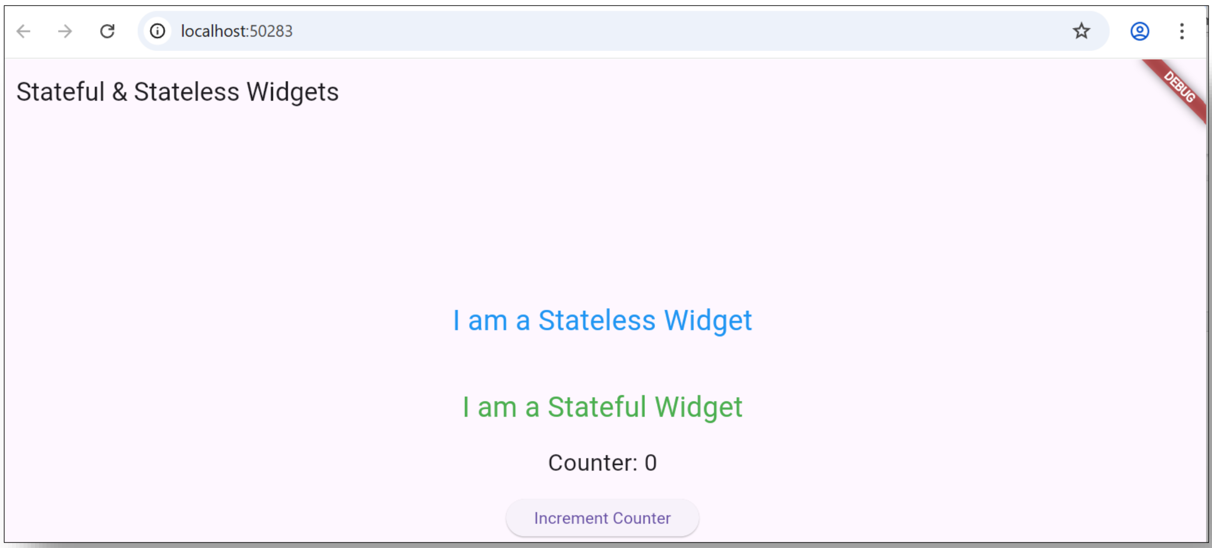
localhost:50283

Stateful & Stateless Widgets

I am a Stateless Widget

I am a Stateful Widget

Counter: 2



localhost:50283

Stateful & Stateless Widgets

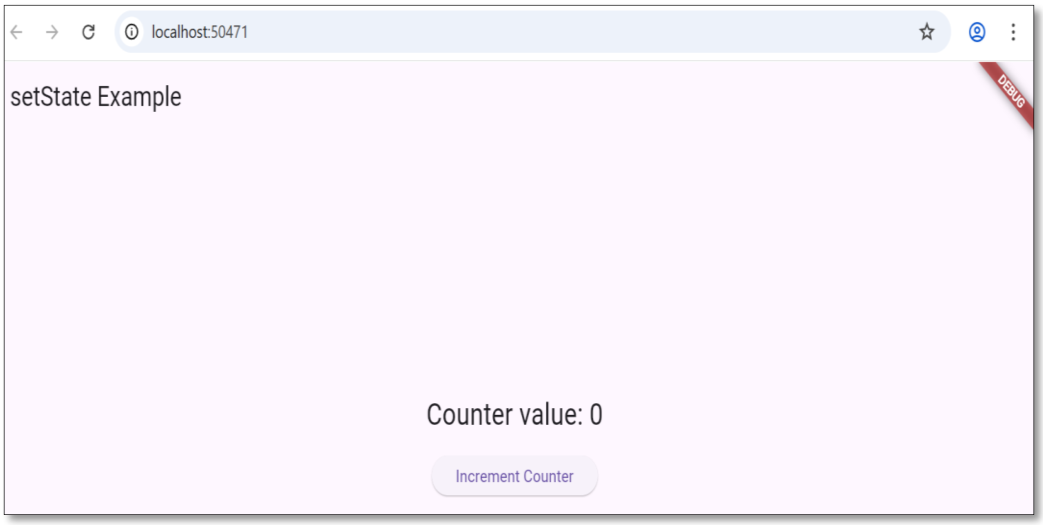
I am a Stateless Widget

I am a Stateful Widget

Counter: 0

5. b) Implement state management using set State and Provider.
Program: using set State

```
import 'package:flutter/material.dart';
void main() {
  runApp(SetStateDemoApp());
}
class SetStateDemoApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'setState Demo',
      home: SetStateCounterScreen(),
    );
  }
}

class SetStateCounterScreen extends StatefulWidget {
  @override
  _SetStateCounterScreenState createState() =>
_SetStateCounterScreenState();
}

class _SetStateCounterScreenState extends
State<SetStateCounterScreen> {
  int _counter = 0;

  void _incrementCounter() {
    setState(() {
      _counter++; // update state
    });
  }
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('setState Example'),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Text('Counter value: $_counter', style: TextStyle(fontSize: 24)),
            SizedBox(height: 16),
            ElevatedButton(
              onPressed: _incrementCounter,
              child: Text('Increment Counter'),
            ),
          ],
        ),
      ),
    );
  }
}
```



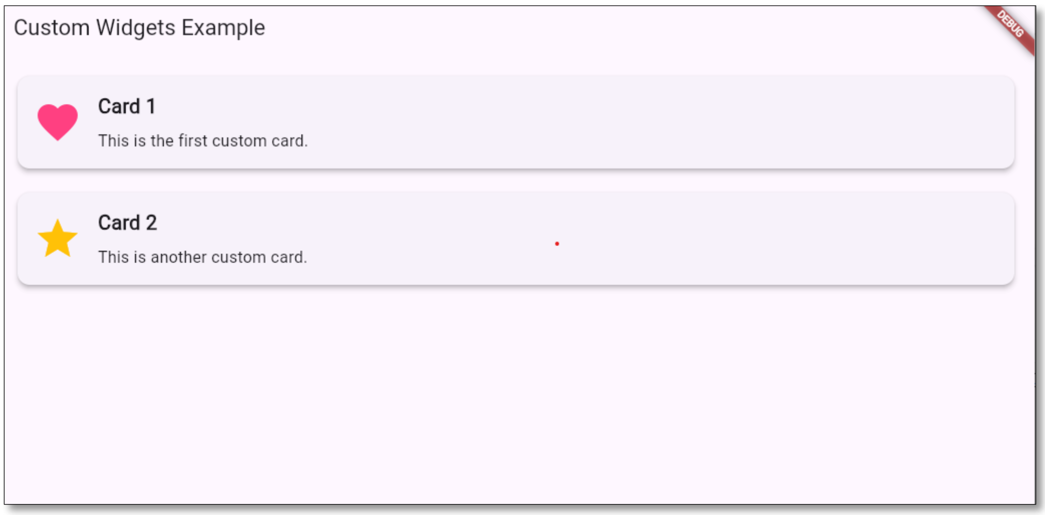
setState Example

Counter value: 0

6a) Create custom widgets for specific UI elements.

```
import 'package:flutter/material.dart';
void main() {
  runApp(CustomWidgetDemoApp());
}
class CustomWidgetDemoApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Custom Widgets Demo',
      home: Scaffold(
        appBar: AppBar(
          title: Text('Custom Widgets Example'),
        ),
        body: Padding(
          padding: const EdgeInsets.all(16.0),
          child: Column(
            children: [
              // Using custom widget
              MyCustomCard(
                title: 'Card 1',
                description: 'This is the first custom card.',
                icon: Icons.favorite,
                color: Colors.pinkAccent,
              ),
              SizedBox(height: 16),
              MyCustomCard(
                title: 'Card 2',
                description: 'This is another custom card.',
                icon: Icons.star,
                color: Colors.amber,
              ),
            ],
          ),
        ),
      ),
    );
  }
}

// Custom Widget: MyCustomCard
class MyCustomCard extends StatelessWidget {
  final String title;
  final String description;
  final IconData icon;
  final Color color;

  // Constructor with named parameters
  const MyCustomCard({
    Key? key,
    required this.title,
    required this.description,
    required this.icon,
    required this.color,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Card(
      elevation: 4,
      shape: RoundedRectangleBorder(borderRadius:
BorderRadius.circular(12)),
      child: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Row(
          children: [
            Icon(icon, size: 48, color: color),
            SizedBox(width: 16),
            Expanded(
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Text(title, style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold)),
                  SizedBox(height: 8),
                  Text(description, style: TextStyle(fontSize: 16)),
                ],
              ),
            ),
          ],
        ),
      ),
    );
```

Output:



Custom Widgets Example

Card 1

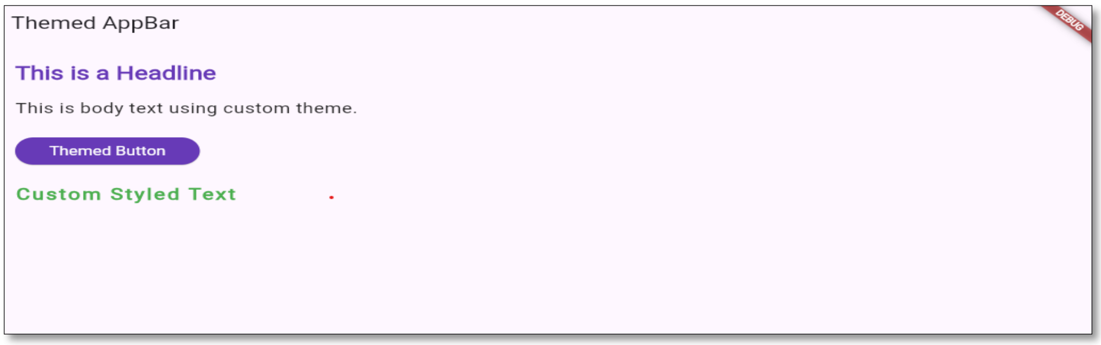
This is the first custom card.

Card 2

This is another custom card.

6b)Apply styling using themes and custom styles.

```
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

// Root Widget
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Theme & Custom Styles Demo',
    // Defining Theme
    theme: ThemeData(
      primarySwatch: Colors.deepPurple,
      brightness: Brightness.light,

      textTheme: TextTheme(
        headlineMedium: TextStyle(fontSize: 24, fontWeight: FontWeight.bold,
color: Colors.deepPurple),
        bodyLarge: TextStyle(fontSize: 18, color: Colors.black87),
      ),

      elevatedButtonTheme: ElevatedButtonThemeData(
        style: ElevatedButton.styleFrom(
          backgroundColor: Colors.deepPurple,
          foregroundColor: Colors.white,
          padding: EdgeInsets.symmetric(horizontal: 32, vertical: 16),
          textStyle: TextStyle(fontSize: 16),
        ),
      ),
    ),

    home: HomeScreen(),
  );
 }
}

// Home Screen Widget
class HomeScreen extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Themed AppBar'),
    ),
    body: Padding(
      padding: const EdgeInsets.all(20.0),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          // Using Theme TextStyle
          Text(
            'This is a Headline',
            style: Theme.of(context).textTheme.headlineMedium,
          ),
          SizedBox(height: 16),
          Text(
            'This is body text using custom theme.',
            style: Theme.of(context).textTheme.bodyLarge,
          ),
          SizedBox(height: 24),

          // Themed Button
          ElevatedButton(
            onPressed: () {
              print('Button pressed!');
            },
            child: Text('Themed Button'),
          ),
          SizedBox(height: 24),

          // Custom Inline Style (Overrides Theme)
          Text(
            'Custom Styled Text',
            style: TextStyle(
              fontSize: 20,
              color: Colors.green,
              fontWeight: FontWeight.w600,
              letterSpacing: 2,
            ),
          ),
        ],
      ),
    ),
  );
 }
}
```


Themed AppBar

This is a Headline

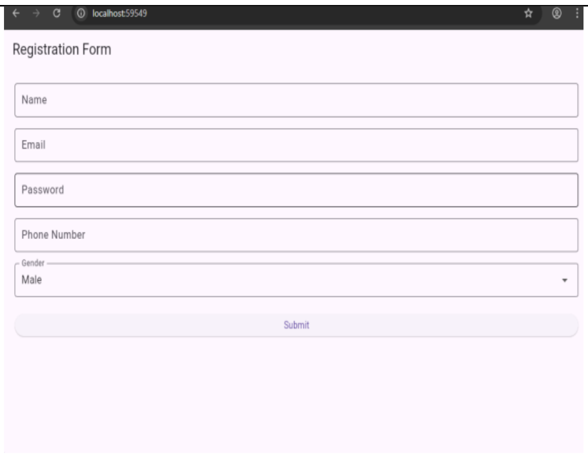
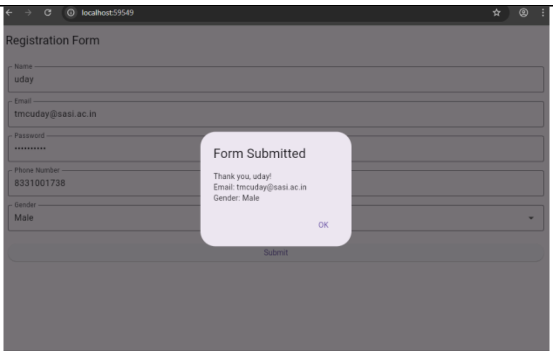
This is body text using custom theme.

Themed Button

Custom Styled Text

7a. Design a form with various input fields

```
import 'package:flutter/material.dart';

void main() => runApp(MaterialApp(home: SimpleForm(),
debugShowCheckedModeBanner: false));

class SimpleForm extends StatefulWidget {
  const SimpleForm({super.key});

  @override
  State<SimpleForm> createState() => _SimpleFormState();
}

class _SimpleFormState extends State<SimpleForm> {
  final _formKey = GlobalKey<FormState>();
  final name = TextEditingController(), email = TextEditingController(), pass =
TextEditingController(), dob = TextEditingController();
  String? gender;
  bool agree = false;

  Future<void> pickDate() async {
    final d = await showDatePicker(context: context, initialDate:
DateTime(2000), firstDate: DateTime(1980), lastDate: DateTime(2030));
    if (d != null) dob.text = "${d.day}-${d.month}-${d.year}";
  }

  void submit() {
    if (_formKey.currentState!.validate() && agree) {
      showDialog(
        context: context,
        builder: (_) => AlertDialog(
          title: Text("Submitted"),
          content: Text("Name: ${name.text}\nEmail: ${email.text}\nGender:
$gender\nDOB: ${dob.text}"),
          actions: [TextButton(onPressed: () => Navigator.pop(context), child:
Text("OK"))],
        ),
      );
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text("Simple Registration")),
      body: Padding(
      padding: const EdgeInsets.all(16),
      child: Form(
        key: _formKey,
        child: ListView(children: [
          TextFormField(controller: name, decoration:
InputDecoration(labelText: "Name"), validator: (v) => v!.isEmpty ? "Enter
name" : null),
          TextFormField(controller: email, decoration: InputDecoration(labelText:
"Email"), validator: (v) => !v!.contains('@') ? "Invalid email" : null),
          TextFormField(controller: pass, decoration: InputDecoration(labelText:
"Password"), obscureText: true, validator: (v) => v!.length < 6 ? "Min 6 chars" :
null),
          DropdownButtonFormField(
            value: gender,
            decoration: InputDecoration(labelText: "Gender"),
            items: ["Male", "Female"].map((g) => DropdownMenuItem(value: g,
child: Text(g))).toList(),
            onChanged: (v) => setState(() => gender = v),
            validator: (v) => v == null ? "Select gender" : null,
          ),
          TextFormField(
            controller: dob,
            readOnly: true,
            decoration: InputDecoration(labelText: "Date of Birth", suffixIcon:
Icon(Icons.calendar_today)),
            onTap: pickDate,
            validator: (v) => v!.isEmpty ? "Select date" : null,
          ),
          CheckboxListTile(
            value: agree,
            onChanged: (v) => setState(() => agree = v!),
            title: Text("I agree to terms"),
            controlAffinity: ListTileControlAffinity.leading,
          ),
          ElevatedButton(onPressed: submit, child: Text("Submit"))
        ]),
      ),
    ),
  );
 }
}
```

 localhost:59549 Registration Form Name Email Password Phone Number Gender: Male Submit	 localhost:59549 Registration Form Name: uday Email: tmcuday@sasi.ac.in Password: •••••••••• Phone Number: 8331001738 Gender: Male Submit Form Submitted Thank you, uday! Email: tmcuday@sasi.ac.in Gender: Male OK

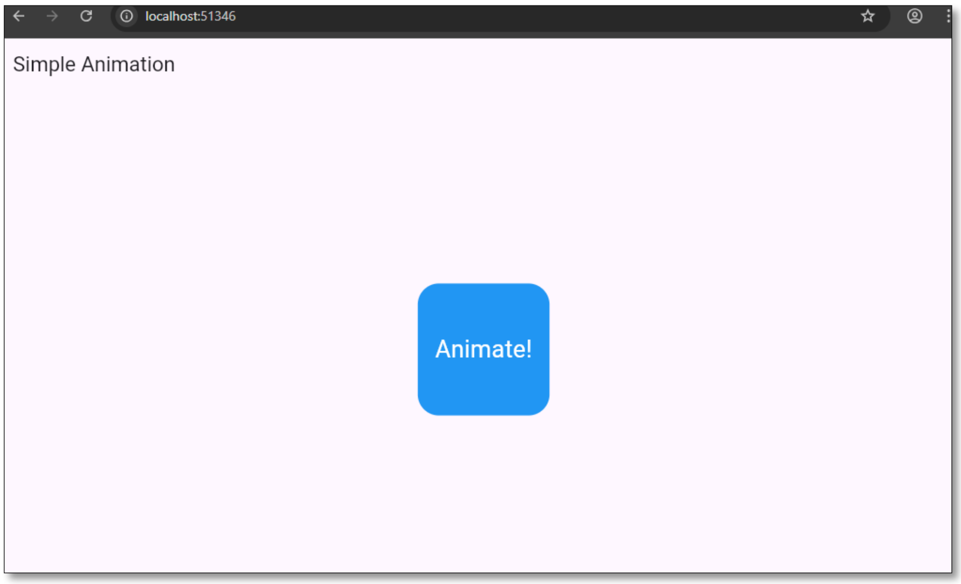
7b. Implement form validation and error handling.

the above example *already includes form validation and error handling*.

Field	Validation Logic	Message Displayed
Name	Checks if empty	"Enter name"
Email	Must contain @	"Invalid email"
Password	Must be at least 6 characters	"Min 6 chars"
Gender (Dropdown)	Must be selected	"Select gender"
Date of Birth	Must not be empty	"Select date"
Checkbox	Must be checked before submitting	SnackBar message: "Please agree to the terms" (handled in logic)

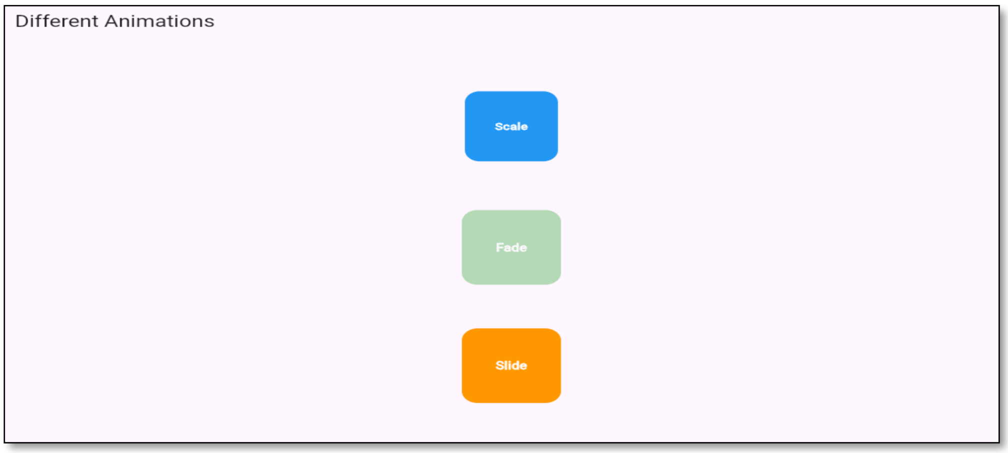
8a. Add animations to UI elements using Flutter's animation framework

```
import 'package:flutter/material.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) =>
      MaterialApp(home: AnimationDemo(),
debugShowCheckedModeBanner: false);
}

class AnimationDemo extends StatefulWidget {
  @override
  State<AnimationDemo> createState() => _AnimationDemoState();
}

class _AnimationDemoState extends State<AnimationDemo>
    with SingleTickerProviderStateMixin {
  late AnimationController controller;
  late Animation<double> scale;

  @override
  void initState() {
    super.initState();
    controller = AnimationController(
      vsync: this,
      duration: Duration(seconds: 2),
    )..repeat(reverse: true);
    scale = Tween(
      begin: 0.5,
      end: 1.5,
    ).animate(CurvedAnimation(parent: controller, curve: Curves.easeInOut));
  }

  @override
  void dispose() {
    controller.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) => Scaffold(
    appBar: AppBar(title: Text("Simple Animation")),
    body: Center(
      child: ScaleTransition(
        scale: scale,
        child: Container(
          width: 100,
          height: 100,
          decoration: BoxDecoration(
            color: Colors.blue,
            borderRadius: BorderRadius.circular(16),
          ),
          child: Center(
            child: Text(
              'Animate!',
              style: TextStyle(color: Colors.white, fontSize: 18),
            ),
          ),
        ),
      ),
    ),
  );
}
```



8b. Experiment with different types of animations (fade, slide, etc.).

```
import 'package:flutter/material.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) =>
      MaterialApp(home: AnimationDemo(),
debugShowCheckedModeBanner: false);
}

class AnimationDemo extends StatefulWidget {
  @override
  State<AnimationDemo> createState() => _AnimationDemoState();
}

class _AnimationDemoState extends State<AnimationDemo>
    with SingleTickerProviderStateMixin {
  late AnimationController controller;
  late Animation<double> scale, fade;
  late Animation<Offset> slide;

  @override
  void initState() {
    super.initState();

    // Controller (2s, repeats back and forth)
    controller = AnimationController(
      vsync: this,
      duration: Duration(seconds: 2),
    )..repeat(reverse: true);

    // Different types of animations
    scale = Tween(
      begin: 0.5,
      end: 1.5,
    ).animate(CurvedAnimation(parent: controller, curve: Curves.easeInOut));

    fade = Tween(
      begin: 0.2,
      end: 1.0,
    ).animate(CurvedAnimation(parent: controller, curve: Curves.easeIn));

    slide = Tween(
      begin: Offset(0, -0.3),
      end: Offset(0, 0.3),
    ).animate(CurvedAnimation(parent: controller, curve: Curves.easeInOut));
  }

  @override
  void dispose() {
    controller.dispose();
    super.dispose();
  }
  Widget buildBox(Color color, String text) => Container(
    width: 100,
    height: 100,
    decoration: BoxDecoration(
      color: color,
      borderRadius: BorderRadius.circular(16),
    ),
    child: Center(
      child: Text(
        text,
        style: TextStyle(color: Colors.white, fontWeight: FontWeight.bold),
      ),
    ),
  );

  @override
  Widget build(BuildContext context) => Scaffold(
    appBar: AppBar(title: Text("Different Animations")),
    body: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.spaceEvenly,
        children: [
          ScaleTransition(scale: scale, child: buildBox(Colors.blue, "Scale")),
          FadeTransition(opacity: fade, child: buildBox(Colors.green, "Fade")),
          SlideTransition(
            position: slide,
            child: buildBox(Colors.orange, "Slide"),
          ),
        ],
      ),
    ),
  );
}
```



9a. Fetch data from a REST API

9b. Display the fetched data in a meaningful way in the UI.

What is REST API?

A REST API is like a bridge that helps your Flutter app talk to another computer called a server. Imagine you are using a food delivery app — when you open it, the app needs to get the list of restaurants, menus, and your orders. The app sends a request to the server through the REST API, and the server sends back the information in a simple format called JSON. REST APIs use the internet just like websites do and follow simple rules. For example, the app uses GET to take information, POST to send new information, PUT to update something, and DELETE to remove something. In Flutter, we can use a package called http to connect to these APIs and show the data on the screen. So, in short, a REST API helps your app get and send data to servers easily, just like talking between two friends who understand the same language.

Pubspec.yaml

Note: don't forget to write your current application name as flutter_appication here "flutter_application_12" is my current one

```
name: flutter_application_12
description: A new Flutter project.

publish_to: "none"

version: 1.0.0+1

environment:
  sdk: '>=3.0.0 <4.0.0'

dependencies:
  flutter:
    sdk: flutter
  http: ^1.2.2

dev_dependencies:
  flutter_test:
    sdk: flutter
  flutter_lints: ^3.0.0

flutter:
  uses-material-design: true
```

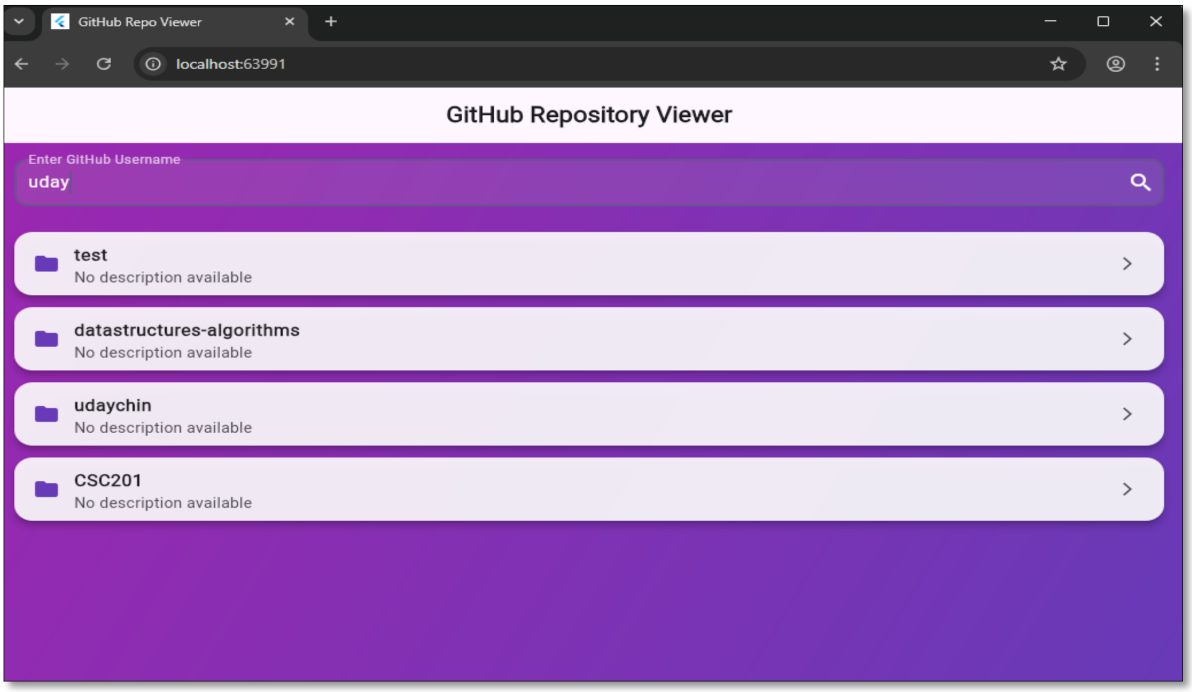
git_hub_api.dart
add new file in "lib" folder of Application directory

```
// lib/git_hub_api.dart
```

```
import 'package:http/http.dart' as http;
import 'dart:convert';

Future<List<dynamic>> fetchRepos(String username) async {
  final url = Uri.parse(
    'https://api.github.com/users/$username/repos?per_page=100',
  );
  final response = await http.get(
    url,
    headers: {'Accept': 'application/vnd.github.v3+json'},
  );

  if (response.statusCode == 200) {
    return json.decode(response.body) as List<dynamic>;
  } else if (response.statusCode == 404) {
    throw Exception('User not found (404)');
  } else {
    throw Exception('Failed to load repos. HTTP ${response.statusCode}');
  }
}
```

main.dart

```
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'dart:convert';

void main() {
  runApp(const GitHubApp());
}

class GitHubApp extends StatelessWidget {
  const GitHubApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'GitHub Repo Viewer',
      theme: ThemeData(primarySwatch: Colors.deepPurple),
      home: const GitHubRepoScreen(),
    );
  }
}

class GitHubRepoScreen extends StatefulWidget {
  const GitHubRepoScreen({super.key});
  @override
  State<GitHubRepoScreen> createState() => _GitHubRepoScreenState();
}

class _GitHubRepoScreenState extends State<GitHubRepoScreen> {
  final TextEditingController _controller = TextEditingController();
  List<dynamic> _repos = [];
  bool _isLoading = false;
  String? _error;

  Future<void> _fetchRepos(String username) async {
    setState(() {
      _isLoading = true;
      _error = null;
      _repos = [];
    });

    final url = Uri.parse('https://api.github.com/users/$username/repos');
    try {
      final response = await http.get(url);
      if (response.statusCode == 200) {
        final data = json.decode(response.body);
        setState(() {
          _repos = List.from(data);
        });
      } else {
        setState(() {
          _error = 'Error: ${response.statusCode} — Failed to load repos';
        });
      }
    } catch (e) {
      setState(() {
        _error = 'An error occurred: $e';
      });
    } finally {
      setState(() {
        _isLoading = false;
      });
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('GitHub Repo Viewer')),
      body: Padding(
    padding: const EdgeInsets.all(16),
    child: Column(
     children: [
      TextField(
       controller: _controller,
       decoration: InputDecoration(
        labelText: 'Enter GitHub Username',
        suffixIcon: IconButton(
         icon: const Icon(Icons.search),
         onPressed: () {
          final username = _controller.text.trim();
          if (username.isNotEmpty) _fetchRepos(username);
         },
        ),
       ),
      ),
      const SizedBox(height: 20),
      if (_isLoading)
       const CircularProgressIndicator()
      else if (_error != null)
       Text(_error!, style: const TextStyle(color: Colors.red))
      else if (_repos.isEmpty)
       const Text('No repositories found.')
      else
       Expanded(
        child: ListView.builder(
         itemCount: _repos.length,
         itemBuilder: (context, index) {
          final repo = _repos[index];
          return ListTile(
           title: Text(repo['name']),
           subtitle: Text(repo['description'] ?? 'No description'),
          );
         },
        ),
       ),
     ],
    ),
   ),
  );
 }
}
```


GitHub Repo Viewer

localhost:63991

GitHub Repository Viewer

Enter GitHub Username

uday

test
No description available

datastructures-algorithms
No description available

udaychin
No description available

CSC201
No description available

10a. Write unit tests for UI components.

Open `lib/main.dart` and see what your root widget is called.

```
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(body: Center(child: Text('Hello Flutter'))),
    );
  }
}
```

Open test/`widget_test.dart`, write:

```
import 'package:flutter_test/flutter_test.dart';
import 'package:flutter_application_14/main.dart'; // ✅correct import

void main() {
  testWidgets('MyApp has Hello Flutter text', (WidgetTester tester) async
{
    await tester.pumpWidget(const MyApp());

    expect(find.text('Hello Flutter'), findsOneWidget);
  });
}
```

Output:

```
PS C:\dw\flutter_application_14> flutter test

00:30 +1: All tests passed!
```

10b. Use Flutter's debugging tools to identify and fix issues.

Flutter provides many **debugging tools** to find and fix UI or logic errors quickly.

1. Flutter DevTools

- Open using:
- flutter pub global activate devtools
- flutter run --debug

  Then open in your browser: http://127.0.0.1:9100

Features:

- **Widget Inspector:** Explore widget tree and properties.
- **Performance Tab:** Check rendering speed, frame drops.
- **Memory Tab:** Detect memory leaks.
- **CPU Profiler:** Analyze slow functions.

2. Debug Mode Shortcuts

- **Hot reload:** r in terminal or Ctrl + \ in VS Code.
- **Hot restart:** Shift + R
- **Debug print:** Use print() or better debugPrint() for logs.

3. Common Debugging Widgets

- **debugPrint()** → prints messages in console.
- **assert()** → checks conditions during debug builds.
- **FlutterError.onError** → catches global Flutter errors.
- **printErrorStackTrace** → shows error with stack trace./

4. Common Tools in IDE

- **VS Code / Android Studio → Run > Start Debugging**
  - Set breakpoints.
  - Inspect variables.
  - Step through code line by line.